Amendments to the Specification

Please change the title of the invention as follows:

--VECTOR TRANSFER SYSTEM GENERATING ADDRESS ERROR EXCEPTION WHEN VECTOR TO BE TRANSFERRED DOES NOT START AND END ON SAME MEMORY PAGE--

On page 1, before "**Field of Invention**" please insert the following paragraph:

The present application is a continuation of U.S. Patent Application Serial No. 10/352,511, filed on January 28, 2003, Attorney Docket No. NEC0217C1US, entitled "Vector Transfer System Generating Address Error Exception When Vector To Be Transferred Does Not Start And End On Same Memory Page", which is a continuation of U.S. Patent Application Serial No. 09/375,873, filed on August 17, 1999, Attorney Docket No. NEC0217US (which issued as U.S. Patent No. 6,513,107 on January 28, 2003) entitled "Vector Transfer System Generating Address Error Exception When Vector To Be Transferred Does Not Start And End On Same Memory Page" both of which are incorporated by reference herein in their entirety and for all purposes.

Please replace the paragraph beginning on page 1, line 21 and ending on page 2, line 14 with the following amended paragraph:

Cache memory attempts to combine the advantages of quick SRAM with the cost efficiency of DRAM to achieve the most effective memory system. Most successive memory accesses affect only a small address area, therefore the most frequently addressed data is held in SRAM cache to provide increased speed over many closely packed memory accesses. Data and code that is not accessed as frequently is stored in slower DRAM. Typically, a memory location is accessed using a row and column within a memory block. A technique known as bursting allows faster memory access when data requested is stored in a contiguous sequence of addresses. During a typical burst, memory is accessed using the starting address, the width of each data element, and the number of data words to access, also referred to as "the stream length". Memory access speed is improved due to the fact there is no need to supply an address for each memory location individually to fetch or store data words from the proper address. One shortfall of this technique arises when data is not stored contiguously in memory, such as when reading or writing an entire row in a matrix since the data is stored by column and then by

row. It is therefore desirable to provide a bursting technique that can accommodate data elements that are not contiguous in memory.

Please add the following paragraph on page 6, lines 15-16:

Figure 8 is a flow chart illustrating steps performed during the process of transferring vector data between a memory and a vector buffer according to an embodiment of the present invention.

Please replace the paragraph on 6, lines 20-25 with the following amended paragraph:

Figure 1 illustrates a computer system **100** which is a simplified example of a computer system with which the present invention may be utilized. It should be noted, however, that the present invention may be utilized in other computer systems having an architecture that is different from computer system **100**. Additionally, the present invention may be implemented in processing systems that do not necessarily include all the features represented in **Figure 1**.

Please replace the paragraph on page 7, lines 1-15 with the following amended paragraph:

Computer system 100 includes processor 102 coupled to host bus 104. External cache memory 106 is also coupled to the host bus 104. Host-to-PCI bridge 108 is coupled to main memory 110, includes cache memory 106 and main memory 110 control functions, and provides bus control to handle transfers among PCI bus 1 12, processor 102, cache memory 106, main memory 110, and host bus 104. PCI bus 112 provides an interface for a variety of devices including, for example, LAN card 114. PCI-to-ISA bridge 116 provides bus control to handle transfers between PCI bus 112 and ISA bus 127, IDE and universal serial bus (USB) functionality 120, and can include other functional elements not shown, such as a real-time clock (RTC), DMA control, interrupt support, and system management bus support. Peripheral devices and input/output (I/O) devices can be attached to various I/O interfaces 122 coupled to ISA bus 127. Alternatively, many I/O devices can be accommodated by a super I/O controller (not shown) attached to ISA bus 127. I/O devices such as modem 124 are coupled to the appropriate I/O interface, for example a serial interface as shown in Figure 1.

Please replace the paragraph on page 7, lines 16-30 with the following amended paragraph:

BIOS 126 is coupled to ISA bus 127, and incorporates the necessary processor executable code for a variety of low-level system functions and system boot functions. BIOS 126 can be stored in any computer readable medium, including magnetic storage media, optical storage media, flash memory, random access memory, read only memory, and communications media conveying signals encoding the instructions (*e.g.* signals from a network). When BIOS 126 boots up (starts up) computer system 100, it first determines whether certain specified hardware in computer system 100 is in place and operating properly. BIOS 126 then loads some or all of operating system 128 from a storage device such as a disk drive into main memory 110. Operating system 128 is a program that manages the resources of computer system 100, such as processor 102, main memory 110, storage device controllers, network interfaces including LAN card 114, various I/O interfaces 122, and data busses 104, 112, 127. Operating system 128 reads one or more configuration files 130 to determine the type and other characteristics of hardware and software resources connected to computer system 100.

Please replace the paragraph on page 8, lines 1-7 with the following amended paragraph:

During operation, main memory 110 includes operating system 128, configuration files 130, and one or more application programs 132 with related program data 134. To increase throughput in computer system 100, program data 134 and instructions from application programs 132 may be placed in cache memory 106 and 136 determined by the pattern of accesses to both data and instructions by the application. Cache memory is typically comprised of SRAM which has relatively fast access time compared to other types of random access memory.

Please replace the paragraph on page 8, lines 8-14 with the following amended paragraph:

As shown in Figures 1 and 2, processor **102** includes internal cache memory **136** and Vector Transfer Unit (VTU) **138**. Internal cache memory **136** is built into processor **102's** circuitry and may be divided functionally into separate instruction caches (I-caches) **202** and data caches (D-caches) **204** where I-cache **202** stores only instructions, and D-cache **204** stores only data. VTU **138** is integrated in processor **102** and includes vector transfer execution unit **206**, vector buffer pool (VBP) **208**, and an efficient bus protocol which supports burst transfers.

Please replace the paragraph on page 12, lines 5-7 with the following amended paragraph:

In this embodiment, bit 11, bit 12, and bits 16 through 29 are currently not utilized. These bits could be used by other embodiments, or to expand capabilities for the present embodiment.

Please replace the paragraph on page 17, lines 11-13 with the following amended paragraph:

R_T is a set of fields including the starting vector buffer address, the length of the vector stream, and the stride of the vector stream.

Please replace the paragraph on page 22, lines 5-10 with the following amended paragraph:

VTU **138** is implemented to execute in parallel with cache memory **136**. On one side, VTU **138** interfaces to memory controller **222**, and on the other side it is connected to the processor core that includes register file **220** and vector transfer execution unit **206**. This configuration achieves high throughput on memory bus **224** by performing vector transfers and executing program instructions using vector data without blocking the pipeline.